The List of Algorithms: Convolutional Layer

1 Convolutional Layer

Convolutional Neural Network is typically consisted with 3 parts. convolutional Layer, pooling layer, fully connected layer. In here, I will focus on convolutional layer, pooling layer will be discussed in other sections.

1.1 Convolution

The classic convolution demonstration we see on tutorials, where a small box moving on a image surface from left to right, is actually a cross-correlation operation. Cross-correlation is a convolution without flipping the kernel, which I don't want to go into details. So, as convention I will address cross-correlation operation a convolution. Here is a 3-D tensor input and 4-D tensor kernel with *(s)tride* parameter convolution.

$$Z_{i,j,k} = c(K, V, s)_{i,j,k} = \sum_{l,m,n} [V_{l,(j-1)\times s+m,(k-1)\times s+n} K_{i,l,m,n}]. \tag{1}$$

Where V is 3-D input, K is 4-D kernel, Z is 3-D output, s is stride.

1.2 Back Propagation

Let a 3-D tensor G such that $G_{i,j,k} = \frac{\partial}{\partial Z_{i,j,k}} J(V, K)$ the gradient we received from back propagation to convolution operation, where $J(V, K)$ is loss function we want to minimize.

To train the convolutional layer, we need the derivatives with respect to the kernel. "To train" means updating kernel K, more specific $K \leftarrow K - \alpha \cdot \nabla_K J$

$$\frac{\partial}{\partial K_{i,j,k,l}} J(V, K) = \sum_{m,n} G_{i,m,n} V_{j,(m-1)\times s+k,(n-1)\times s+l}. \tag{2}$$

To pass the gradient through back-propagation to next layer, we need to compute gradient respect to V.

$$\frac{\partial}{\partial V_{i,j,k}} J(V,K) = \sum_{\substack{l,m \\ s.t. \\ (l-1)\times s+m=j}} \sum_{\substack{n,p \\ s.t. \\ (n-1)\times s+p=k}} \sum_{q} K_{q,i,m,p} G_{q,l,n}. \tag{3}$$

Here is an example. Let's $V \in \mathbb{R}^{3\times 3}, K \in \mathbb{R}^{2\times 2}, Z \in \mathbb{R}^{2\times 2}, G \in \mathbb{R}^{2\times 2}, s = 1$.

$$V = \begin{bmatrix} V_{1,1} & V_{1,2} & V_{1,3} \\ V_{2,1} & V_{2,2} & V_{2,3} \\ V_{3,1} & V_{3,2} & V_{3,3} \end{bmatrix}, K = \begin{bmatrix} K_{1,1} & K_{1,2} \\ K_{2,1} & K_{2,2} \end{bmatrix}, Z = \begin{bmatrix} Z_{1,1} & Z_{1,2} \\ Z_{2,1} & Z_{2,2} \end{bmatrix}, G = \begin{bmatrix} G_{1,1} & G_{1,2} \\ G_{2,1} & G_{2,2} \end{bmatrix}.$$

where

$$\begin{aligned} Z_{1,1} &= K_{1,1} \times V_{1,1} + K_{1,2} \times V_{1,2} + K_{2,1} \times V_{2,1} + K_{2,2} \times V_{2,2}, \\ Z_{1,2} &= K_{1,1} \times V_{1,2} + K_{1,2} \times V_{1,3} + K_{2,1} \times V_{2,2} + K_{2,2} \times V_{2,3}, \\ Z_{2,1} &= K_{1,1} \times V_{2,1} + K_{1,2} \times V_{2,2} + K_{2,1} \times V_{3,1} + K_{2,2} \times V_{3,2}, \\ Z_{2,2} &= K_{1,1} \times V_{2,2} + K_{1,2} \times V_{2,3} + K_{2,1} \times V_{3,2} + K_{2,2} \times V_{3,3}. \end{aligned}$$

then, partial derivative with respect to $K_{1,1}$ is

$$\begin{aligned} \frac{\partial}{\partial K_{1,1}} J(V,K) &= \frac{\partial Z}{\partial K_{1,1}} \frac{\partial J(V,K)}{\partial Z} \\ &= \frac{\partial Z_{1,1}}{\partial K_{1,1}} \frac{\partial J}{\partial Z_{1,1}} + \frac{\partial Z_{1,2}}{\partial K_{1,1}} \frac{\partial J}{\partial Z_{1,2}} + \frac{\partial Z_{2,1}}{\partial K_{1,1}} \frac{\partial J}{\partial Z_{2,1}} + \frac{\partial Z_{2,2}}{\partial K_{1,1}} \frac{\partial J}{\partial Z_{2,2}} \\ &= V_{1,1} \cdot G_{1,1} + V_{1,2} \cdot G_{1,2} + V_{2,1} \cdot G_{2,1} + V_{2,2} \cdot G_{2,2}. \end{aligned}$$

Partial derivative respect to $V_{1,1}$ is

$$\begin{aligned} \frac{\partial}{\partial V_{1,1}} J(V,K) &= \frac{\partial Z}{\partial V_{1,1}} \frac{\partial J(V,K)}{\partial Z} \\ &= \frac{\partial Z_{1,1}}{\partial V_{1,1}} \frac{\partial J}{\partial Z_{1,1}} + \frac{\partial Z_{1,2}}{\partial V_{1,1}} \frac{\partial J}{\partial Z_{1,2}} + \frac{\partial Z_{2,1}}{\partial V_{1,1}} \frac{\partial J}{\partial Z_{2,1}} + \frac{\partial Z_{2,2}}{\partial V_{1,1}} \frac{\partial J}{\partial Z_{2,2}} \\ &= K_{1,1} \cdot G_{1,1}. \end{aligned}$$

Partial derivative with respect to $V_{2,2}$ is

$$\begin{aligned} \frac{\partial}{\partial V_{2,2}} J(V,K) &= \frac{\partial Z}{\partial V_{2,2}} \frac{\partial J(V,K)}{\partial Z} \\ &= \frac{\partial Z_{1,1}}{\partial V_{2,2}} \frac{\partial J}{\partial Z_{1,1}} + \frac{\partial Z_{1,2}}{\partial V_{2,2}} \frac{\partial J}{\partial Z_{1,2}} + \frac{\partial Z_{2,1}}{\partial V_{2,2}} \frac{\partial J}{\partial Z_{2,1}} + \frac{\partial Z_{2,2}}{\partial V_{2,2}} \frac{\partial J}{\partial Z_{2,2}} \\ &= K_{2,2} \cdot G_{1,1} + K_{2,1} \cdot G_{1,2} + K_{1,2} \cdot G_{2,1} + K_{1,1} \cdot G_{2,2}. \end{aligned}$$

Here, Let's see how $\sum_{n,ps.t.(n-1)\times s+p=k}$ works in equation 3. Let's isolate out k, p, n from equation.

$$\frac{\partial J}{\partial V_k} = \sum_{\substack{n,p \\ s.t. \\ (n-1)\times s+p=k}} K_p G_n.$$

Where p is changing between 1 and kernel size. k is given. n is constraint to $(n-1) \times s + p = k$. If k is given and p is fixed, then n is fixed too. For given $k = 1$, when $p = 1$, then $n = 1$, when $p = 2$, there is no n satisfies constraint, so

$$\frac{\partial J}{\partial V_1} = K_1 G_1.$$

For given $k = 2$, when $p = 1$, then $n = 2$, when $p = 2$, $n = 1$, so

$$\frac{\partial J}{\partial V_2} = K_1 \cdot G_2 + K_2 \cdot G_1.$$

The reason why I use $\frac{\partial}{\partial V_{1,1}} J(V, K)$ and $\frac{\partial}{\partial V_{2,2}} J(V, K)$, NOT $\frac{\partial}{\partial V_{1,2}} J(V, K)$ is x axis, y axis, z axis is not in conventional order. i.e., in $V_{i,j,k}$ i is channel, j is x axis (rows), k is y axis (channel) for image; in $K_{i,j,k,m}$ i is output channel, j is input channel, k is rows, l is columns. Simply, conventional order is $A_{x,y,z}$, in here it's $A_{z,x,y}$.

2 More

So, what does 2D mean in 2D convolution? It means the movement of kernels, not subject. means, when doing on mnist image and cifar 10 image, which is difference between 1 channel and 3 channels, the convolution we apply is actually same math. seems like mnist is 2D and cifar10 is 3D, isn't it?. if you think about next convolutional layer that connected to previous convolutional layer that has 16 channels outputs. it makes sense isn't it.

References

[1] Deep Learning Book.